

# INTERRUPT

Home

## Apps

Here is a list of all modules:

- License Terms and Copyright Information
- Abbreviations and Definitions
- Overview
- Architecture Description
- APP Configuration Parameters
- Enumerations
- Data structures
- Methods
- Usage
- Release History
- INTERRUPT

# INTERRUPT

Home

## License Terms and Copyright Information

### License Terms and Copyright Information

Copyright (c) 2015, Infineon Technologies AG All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

To improve the quality of the software, users are encouraged to share modifications, enhancements or bug fixes with Infineon Technologies AG ([dave@infineon.com](mailto:dave@infineon.com)).

---

# INTERRUPT

Home

## Abbreviations and Definitions

### Abbreviations and Definitions

Abbreviations:	
DAVE™	Digital Application Virtual Engineer
APP	DAVE Application
API	Application Programming Interface
GUI	Graphical User Interface
MCU	Microcontroller Unit
SW	Software
HW	Hardware
LLD	Low Level Driver
SCU	System Control Unit
IO	Input Output
NVIC	Nested Vector Interrupt Controller

Definitions:	
Singleton	Only single instance of the APP is permitted
Sharable	Resource sharing with other APPs is permitted
initProvider	Provides the initialization routine
Physical connectivity	Hardware inter/intra peripheral (constant) signal connection
Conditional connectivity	Constrained hardware inter/intra peripheral signal connection
Aggregation	Indicates consumption of low level (dependent) APPs



# INTERRUPT

Home

## Overview

### Overview

The **INTERRUPT** APP is a system APP. The Cortex-M vector table contains the address of the exception handlers and interrupts service routine (ISR). It allows the user to overwrite the provided default implementation of the interrupt service routine and to set the interrupt priority.

The user needs to provide the implementation of the ISR. The user has also the choice to enable the interrupt at initialization.

The **INTERRUPT** APP requires the CPU APP to be informed about the number of priority levels and in case of Cortex-M4 also the number of subpriority levels.

The **INTERRUPT** APP based on the peripheral service request connectivity resolves the NVIC IRQ node to be used.

### Supported Devices

1. XMC4800/XMC4700 Series
2. XMC4500 Series
3. XMC4400 Series
4. XMC4300 Series
5. XMC4200 / XMC4100 Series
6. XMC1400 Series
7. XMC1300 Series
8. XMC1200 Series
9. XMC1100 Series

### References

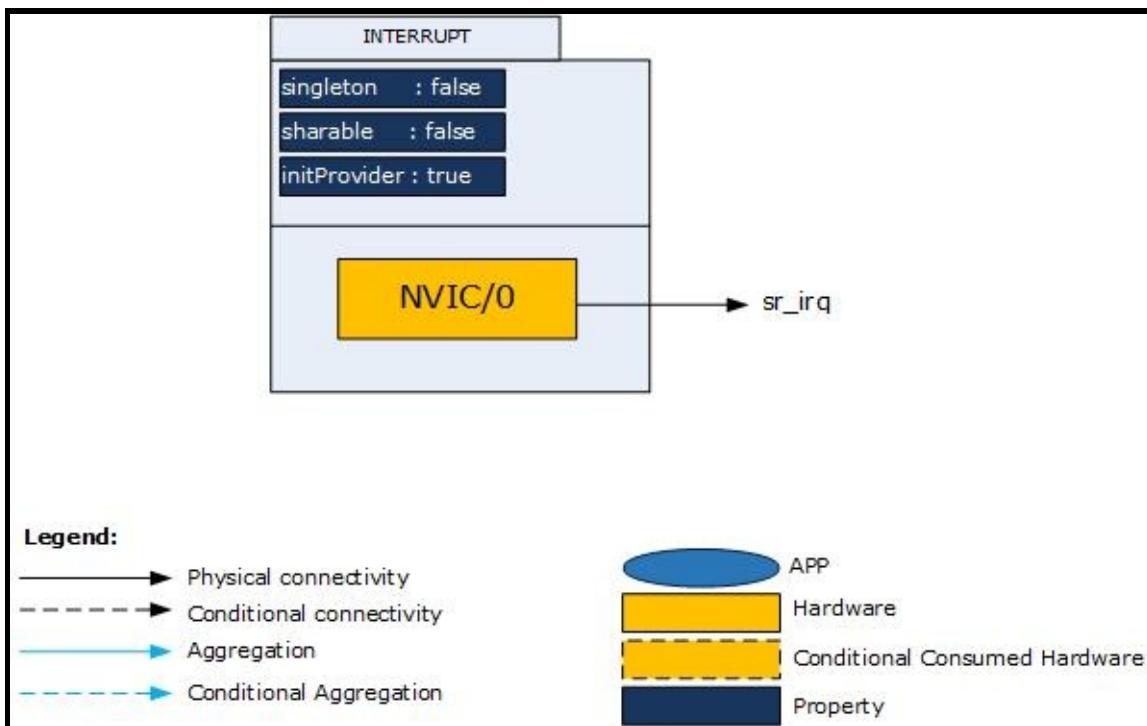
1. XMC4800/XMC4700 Reference Manual
  2. XMC4500 Reference Manual
  3. XMC4400 Reference Manual
  4. XMC4300 Reference Manual
  5. XMC4200 / XMC4100 Reference Manual
  6. XMC1400 Reference Manual
  7. XMC1300 Reference Manual
  8. XMC1200 Reference Manual
  9. XMC1100 Reference Manual
-

# INTERRUPT

Home

## Architecture Description

### Architecture Description



**Figure 1 :** Architecture of **INTERRUPT** APP

The above diagram represents the internal software architecture of the **INTERRUPT** APP and its instance exists in a DAVE™ project with fixed attributes as shown. Each instance of this APP consumes one NVIC node in the MCU. The **INTERRUPT** APP also provides input signal for inter-peripheral connections.

An instantiated APP generates (after code generation) a specific data structure with the GUI configuration. The name of this data structure can be modified by changing the APP instance label (e.g. change label from default **INTERRUPT\_0** to **MY\_INTERRUPT**).

## Signals:

The following table describes the list of IO signals for **INTERRUPT APP**.

**Table 1:** APP Input Output signals

Signal Name	Input/Output	Availability	Description
sr_irq	Input	Always	This signal can be used to connect to any other source which can generate interrupt.

# INTERRUPT

Home

## APP Configuration Parameters

### App Configuration Parameters

Interrupt Settings

Enable interrupt at initialization

Interrupt Priority

Preemption priority 63 Subpriority 0

Interrupt handler: UserIRQHandler

Figure 1: Interrupt Settings

# INTERRUPT

Home

## Enumerations

```
INTERRUPT_STATUS {  
    INTERRUPT_STATUS_SUCCESS = 0U,  
    INTERRUPT_STATUS_FAILURE = 1U }  
typedef enum INTERRUPT_STATUS INTERRUPT_STATUS_t
```

## Typedef Documentation

`typedef enum INTERRUPT_STATUS INTERRUPT_STATUS_t`

Initialization status.

## Enumeration Type Documentation

### enum INTERRUPT\_STATUS

Initialization status.

#### Enumerator:

*INTERRUPT\_STATUS\_SUCCESS* APP initialization success

*INTERRUPT\_STATUS\_FAILURE* APP initialization failure

Definition at line **94** of file **INTERRUPT.h**.

# INTERRUPT

Home

Data Structures

**Data structures**

## Data Structures

struct **INTERRUPT**

This structure holds run-time configurations of **INTERRUPT APP**.  
[More...](#)

typedef struct **INTERRUPT** **INTERRUPT\_t**

This structure holds run-time configurations of **INTERRUPT APP**.

---



# INTERRUPT

Home

## Methods

DAVE\_APP\_VERSION\_t **INTERRUPT\_GetAppVersion** (void)

Get INTERRUPT APP version.

**INTERRUPT\_STATUS\_t** **INTERRUPT\_Init** (const INTERRUPT\_t \*const handler)

Initializes INTERRUPT APP instance.

\_\_STATIC\_INLINE void

**INTERRUPT\_Enable** (const INTERRUPT\_t \*const handler)

Enables the IRQ.

\_\_STATIC\_INLINE void

**INTERRUPT\_Disable** (const INTERRUPT\_t \*const handler)

Disables the IRQ.

\_\_STATIC\_INLINE uint32\_t

**INTERRUPT\_GetPending** (const INTERRUPT\_t \*const handler)

Get the pending IRQ.

\_\_STATIC\_INLINE void

**INTERRUPT\_SetPending** (const INTERRUPT\_t \*const handler)

Set the IRQ to pending state.

\_\_STATIC\_INLINE void

**INTERRUPT\_ClearPending** (const INTERRUPT\_t \*const handler)

Clears the pending status of the IRQ.

\_\_STATIC\_INLINE uint32\_t

**INTERRUPT\_GetActive** (const INTERRUPT\_t \*const handler)

Get current running active status of the IRQ. This API is applicable only for XMC4000 devices.

## Methods

## Function Documentation

**`__STATIC_INLINE void INTERRUPT_ClearPending ( const INTERRUPT_t * handle )`**

Clears the pending status of the IRQ.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate two instances of **INTERRUPT** APP

```
#include <DAVE.h>

uint32_t pend_IRQ;
int main(void)
{
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    INTERRUPT_Enable(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}

void MyISR_handler(void)
{
    INTERRUPT_Enable(&INTERRUPT_1);
    INTERRUPT_SetPending(&INTERRUPT_1);
    pend_IRQ = INTERRUPT_GetPending(&INTERRUPT_1)
;
    if(pend_IRQ)
```

```
{  
    INTERRUPT_Disable(&INTERRUPT_0);  
    INTERRUPT_ClearPending(&INTERRUPT_1);  
}  
}
```

Definition at line 324 of file **INTERRUPT.h**.

References **INTERRUPT::node**.

---

```
_STATIC_INLINE void INTERRUPT_Disable ( const INTERRUPT_t *
```

Disables the IRQ.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT APP**

```
#include <DAVE.h>  
  
int main(void)  
{  
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()  
    INTERRUPT_Disable(&INTERRUPT_0);  
    while(1)  
    {}  
    return 0;
```

```
}
```

Definition at line **235** of file **INTERRUPT.h**.

References **INTERRUPT::node**.

---

```
__STATIC_INLINE void INTERRUPT_Enable ( const INTERRUPT_t *
```

Enables the IRQ.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT APP**

```
#include <DAVE.h>

int main(void)
{
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    INTERRUPT_Enable(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line **210** of file **INTERRUPT.h**.

References [INTERRUPT::node](#).

Referenced by [INTERRUPT\\_Init\(\)](#).

## **`__STATIC_INLINE uint32_t INTERRUPT_GetActive ( const INTERRUPT_t * handle )`**

Get current running active status of the IRQ. This API is applicable only for XMC4000 devices.

**Parameters:**

**handle** Constant pointer to constant structure of type  
[INTERRUPT\\_t](#)

**Returns:**

`uint32_t` current active running IRQ node

**Example:** Pre-requisite: Instantiate one instance of [INTERRUPT](#) APP

```
#include <DAVE.h>

int main(void)
{
    uint32_t Status;
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    Status = INTERRUPT_GetActive(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line 352 of file [INTERRUPT.h](#).

References [INTERRUPT::node](#).

## DAVE\_APP\_VERSION\_t INTERRUPT\_GetAppVersion ( void )

Get INTERRUPT APP version.

### Returns:

DAVE\_APP\_VERSION\_t APP version information (major, minor and patch number)

### Description:

The function can be used to check application software compatibility with a specific version of the APP.

```
#include <DAVE.h>

int main(void)
{
    DAVE_APP_VERSION_t version;
    DAVE_Init();
    version = INTERRUPT_GetAppVersion();
    if(version.major != 4U)
    {
    }
    while(1)
    {}
    return 0;
}
```

Definition at line 79 of file INTERRUPT.c.

## \_\_STATIC\_INLINE uint32\_t INTERRUPT\_GetPending ( const INTERI

Get the pending IRQ.

### Parameters:

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

uint32\_t IRQ node

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT APP**

```
#include <DAVE.h>

int main(void)
{
    uint32_t Status;
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    Status = INTERRUPT_GetPending(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line **261** of file **INTERRUPT.h**.

References **INTERRUPT::node**.

**INTERRUPT\_STATUS\_t INTERRUPT\_Init ( const INTERRUPT\_t \*cor**

Initializes **INTERRUPT APP** instance.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

**INTERRUPT\_STATUS\_t**

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT\_APP**

```
#include <DAVE.h>

int main(void)
{
    DAVE_Init(); // INTERRUPT_Init(&INTERRUPT_0)
is called within DAVE_Init()
    while(1)
    {}
    return 0;
}
```

Definition at line **93** of file **INTERRUPT.c**.

References **INTERRUPT::enable\_at\_init**, **INTERRUPT\_Enable()**, **INTERRUPT\_STATUS\_SUCCESS**, **INTERRUPT::irqctrl**, **INTERRUPT::node**, **INTERRUPT::priority**, and **INTERRUPT::subpriority**.

**\_\_STATIC\_INLINE void INTERRUPT\_SetPending ( const INTERRUPT\_t \* handle )**

Set the IRQ to pending state.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT** APP

```
#include <DAVE.h>

int main(void)
{
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    INTERRUPT_SetPending(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line **286** of file **INTERRUPT.h**.

References **INTERRUPT::node**.

---

# INTERRUPT

Home

## Usage

### Usage

**INTERRUPT** is a global DAVE™ APP. It is used by other APPs which required interrupt connectivity. For information on how **INTERRUPT** is being used, refer for example to the ERU related APPs help documentation.

---

# INTERRUPT

Home

## Release History

### Release History



# INTERRUPT

Home

Data Structures

**INTERRUPT**

## Data Structures

struct	<b>INTERRUPT</b>
	This structure holds run-time configurations of <b>INTERRUPT</b> APP. More...
<b>INTERRUPT_STATUS_t</b>	<b>INTERRUPT_Init</b> (const <b>INTERRUPT_t</b> *const handler) Initializes <b>INTERRUPT</b> APP instance.
<b>_STATIC_INLINE void</b>	<b>INTERRUPT_Enable</b> (const <b>INTERRUPT_t</b> *const handler) Enables the IRQ.
<b>_STATIC_INLINE void</b>	<b>INTERRUPT_Disable</b> (const <b>INTERRUPT_t</b> *const handler) Disables the IRQ.
<b>_STATIC_INLINE uint32_t</b>	<b>INTERRUPT_GetPending</b> (const <b>INTERRUPT_t</b> *const handler) Get the pending IRQ.
<b>_STATIC_INLINE void</b>	<b>INTERRUPT_SetPending</b> (const <b>INTERRUPT_t</b> *const handler) Set the IRQ to pending state.
<b>_STATIC_INLINE void</b>	<b>INTERRUPT_ClearPending</b> (const <b>INTERRUPT_t</b> *const handler) Clears the pending status of the IRQ.
<b>_STATIC_INLINE uint32_t</b>	<b>INTERRUPT_GetActive</b> (const <b>INTERRUPT_t</b> *const handler) Get current running active status of the IRQ. This API is applicable only for XMC4000 devices.

## Function Documentation

**\_\_STATIC\_INLINE void INTERRUPT\_ClearPending ( const INTERRUPT\_t \* handle )**

Clears the pending status of the IRQ.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate two instances of **INTERRUPT** APP

```
#include <DAVE.h>

uint32_t pend_IRQ;
int main(void)
{
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    INTERRUPT_Enable(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}

void MyISR_handler(void)
{
    INTERRUPT_Enable(&INTERRUPT_1);
    INTERRUPT_SetPending(&INTERRUPT_1);
    pend_IRQ = INTERRUPT_GetPending(&INTERRUPT_1)
;
    if(pend_IRQ)
```

```
{  
    INTERRUPT_Disable(&INTERRUPT_0);  
    INTERRUPT_ClearPending(&INTERRUPT_1);  
}  
}
```

Definition at line 324 of file **INTERRUPT.h**.

References **node**.

**\_\_STATIC\_INLINE void INTERRUPT\_Disable ( const INTERRUPT\_t \* handle )**

Disables the IRQ.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT APP**

```
#include <DAVE.h>  
  
int main(void)  
{  
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()  
    INTERRUPT_Disable(&INTERRUPT_0);  
    while(1)  
    {}  
    return 0;
```

```
}
```

Definition at line **235** of file **INTERRUPT.h**.

References **node**.

---

```
_STATIC_INLINE void INTERRUPT_Enable ( const INTERRUPT_t *
```

Enables the IRQ.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT\_APP**

```
#include <DAVE.h>

int main(void)
{
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    INTERRUPT_Enable(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line **210** of file **INTERRUPT.h**.

References [node](#).

Referenced by [INTERRUPT\\_Init\(\)](#).

## **`__STATIC_INLINE uint32_t INTERRUPT_GetActive ( const INTERRUPT_t * handle )`**

Get current running active status of the IRQ. This API is applicable only for XMC4000 devices.

**Parameters:**

**handle** Constant pointer to constant structure of type  
[INTERRUPT\\_t](#)

**Returns:**

`uint32_t` current active running IRQ node

**Example:** Pre-requisite: Instantiate one instance of [INTERRUPT\\_APP](#)

```
#include <DAVE.h>

int main(void)
{
    uint32_t Status;
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    Status = INTERRUPT_GetActive(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line 352 of file [INTERRUPT.h](#).

References [node](#).

## **\_\_STATIC\_INLINE uint32\_t INTERRUPT\_GetPending ( const INTERRUPT\_t \*cor**

Get the pending IRQ.

### **Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

### **Returns:**

uint32\_t IRQ node

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT\_APP**

```
#include <DAVE.h>

int main(void)
{
    uint32_t Status;
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    Status = INTERRUPT_GetPending(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line **261** of file **INTERRUPT.h**.

References **node**.

## **INTERRUPT\_STATUS\_t INTERRUPT\_Init ( const INTERRUPT\_t \*cor**

Initializes **INTERRUPT** APP instance.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

**INTERRUPT\_STATUS\_t**

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT** APP

```
#include <DAVE.h>

int main(void)
{
    DAVE_Init(); // INTERRUPT_Init(&INTERRUPT_0)
is called within DAVE_Init()
    while(1)
    {}
    return 0;
}
```

Definition at line 93 of file **INTERRUPT.c**.

References **enable\_at\_init**, **INTERRUPT\_Enable()**,  
**INTERRUPT\_STATUS\_SUCCESS**, **irqctrl**, **node**, **priority**, and  
**subpriority**.

**\_\_STATIC\_INLINE void INTERRUPT\_SetPending ( const INTERRUPT\_t \* handle, uint8\_t priority, uint8\_t subpriority )**

Set the IRQ to pending state.

**Parameters:**

**handle** Constant pointer to constant structure of type  
**INTERRUPT\_t**

**Returns:**

None

**Example:** Pre-requisite: Instantiate one instance of **INTERRUPT\_APP**

```
#include <DAVE.h>

int main(void)
{
    DAVE_Init(); // INTERRUPT_Init() is called within DAVE_Init()
    INTERRUPT_SetPending(&INTERRUPT_0);
    while(1)
    {}
    return 0;
}
```

Definition at line 286 of file **INTERRUPT.h**.

References **node**.

# INTERRUPT

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

## Data Structures

Here are the data structures with brief descriptions:

**INTERRUPT**

This structure holds run-time configurations of  
**INTERRUPT APP**

---

---

# INTERRUPT

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

Data Fields

## INTERRUPT Struct Reference

[INTERRUPT | Data structures](#)

## Detailed Description

This structure holds run-time configurations of **INTERRUPT APP**.

Definition at line **114** of file **INTERRUPT.h**.

```
#include <INTERRUPT.h>
```

## Data Fields

```
const XMC_SCU_IRQCTRL_t irqctrl
    const IRQn_Type node
    const uint8_t priority
    const uint8_t subpriority
    const bool enable_at_init
```

---

## Field Documentation

### `const bool INTERRUPT::enable_at_init`

Interrupt enable for Node

Definition at line [124](#) of file **INTERRUPT.h**.

Referenced by **INTERRUPT\_Init()**.

### `const XMC_SCU_IRQCTRL_t INTERRUPT::irqctrl`

selects the interrupt source for a NVIC interrupt node

Definition at line [117](#) of file **INTERRUPT.h**.

Referenced by **INTERRUPT\_Init()**.

### `const IRQn_Type INTERRUPT::node`

Mapped NVIC Node

Definition at line [119](#) of file **INTERRUPT.h**.

Referenced by **INTERRUPT\_ClearPending()**,  
**INTERRUPT\_Disable()**, **INTERRUPT\_Enable()**,  
**INTERRUPT\_GetActive()**, **INTERRUPT\_GetPending()**,  
**INTERRUPT\_Init()**, and **INTERRUPT\_SetPending()**.

### `const uint8_t INTERRUPT::priority`

Node Interrupt Priority

Definition at line [120](#) of file **INTERRUPT.h**.

Referenced by [INTERRUPT\\_Init\(\)](#).

### **const uint8\_t INTERRUPT::subpriority**

Node Interrupt SubPriority only valid for XMC4x

Definition at line [122](#) of file [INTERRUPT.h](#).

Referenced by [INTERRUPT\\_Init\(\)](#).

---

The documentation for this struct was generated from the following file:

- [INTERRUPT.h](#)
-

# INTERRUPT

Home

Data Structures

Data Structure Index

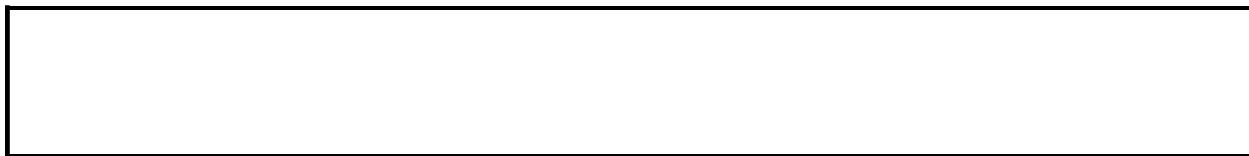
Data Fields

## Data Structure Index

I

INTERRUPT

I



# INTERRUPT

Home			
Data Structures	Data Structure Index		Data Fields
All	Variables		

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- enable\_at\_init : [INTERRUPT](#)
  - irqctrl : [INTERRUPT](#)
  - node : [INTERRUPT](#)
  - priority : [INTERRUPT](#)
  - subpriority : [INTERRUPT](#)
- 



# INTERRUPT

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

- enable\_at\_init : INTERRUPT
  - irqctrl : INTERRUPT
  - node : INTERRUPT
  - priority : INTERRUPT
  - subpriority : INTERRUPT
- 
-

# INTERRUPT

Home

File List

Globals

## File List

Here is a list of all documented files with brief descriptions:

[INTERRUPT.c](#) [code] 

[INTERRUPT.h](#) [code] 

---

# INTERRUPT

[Home](#)

[File List](#)

[Globals](#)

[Functions](#)

## INTERRUPT.c File Reference

## Detailed Description

**Date:**

2015-09-18

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file **INTERRUPT.c**.

```
#include "interrupt.h"
```

## Functions

DAVE_APP_VERSION_t	<b>INTERRUPT_GetAppVersion</b> (void)
Get <b>INTERRUPT</b> APP version.	
<b>INTERRUPT_STATUS_t</b>	<b>INTERRUPT_Init</b> (const <b>INTERRUPT_t</b> *const handler)
Initializes <b>INTERRUPT</b> APP instance.	

Go to the source code of this file.

---

---

# INTERRUPT

[Home](#)

[File List](#)

[Globals](#)

[Data Structures](#)

## INTERRUPT.h File Reference

## Detailed Description

**Date:**

2015-10-05

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file **INTERRUPT.h**.

```
#include <xmc_common.h> #include <DAVE_Common.h>
#include <xmc_scu.h>
#include "interrupt_conf.h"
#include "interrupt_extern.h"
```

## Data Structures

---

struct **INTERRUPT**

This structure holds run-time configurations of **INTERRUPT** APP. More...

## TypeDefs

---

```
typedef struct INTERRUPT INTERRUPT_t
```

This structure holds run-time  
configurations of **INTERRUPT APP.**

## Functions

DAVE_APP_VERSION_t	<b>INTERRUPT_GetAppVersion</b> (void) Get <b>INTERRUPT</b> APP version.
<b>INTERRUPT_STATUS_t</b>	<b>INTERRUPT_Init</b> (const <b>INTERRUPT_t</b> *const handler) Initializes <b>INTERRUPT</b> APP instance.
<u>__STATIC_INLINE</u> void	<b>INTERRUPT_Enable</b> (const <b>INTERRUPT_t</b> *const handler) Enables the IRQ.
<u>__STATIC_INLINE</u> void	<b>INTERRUPT_Disable</b> (const <b>INTERRUPT_t</b> *const handler) Disables the IRQ.
<u>__STATIC_INLINE</u> uint32_t	<b>INTERRUPT_GetPending</b> (cons <b>INTERRUPT_t</b> *const handler) Get the pending IRQ.
<u>__STATIC_INLINE</u> void	<b>INTERRUPT_SetPending</b> (cons <b>INTERRUPT_t</b> *const handler) Set the IRQ to pending state.
<u>__STATIC_INLINE</u> void	<b>INTERRUPT_ClearPending</b> (const <b>INTERRUPT_t</b> *const handler) Clears the pending status of the IRQ.
<u>__STATIC_INLINE</u> uint32_t	<b>INTERRUPT_GetActive</b> (const <b>INTERRUPT_t</b> *const handler) Get current running active status of the IRQ. This API is applicable only for XMC4000 devices.
	<b>INTERRUPT_STATUS {</b> <b>INTERRUPT_STATUS_SUCCESS</b>

```
enum = 0U,  
INTERRUPT_STATUS_FAILURE  
= 1U }  
typedef enum INTERRUPT_STATUS INTERRUPT_STATUS_t
```

Go to the source code of this file.

---



# INTERRUPT

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- INTERRUPT\_ClearPending() : [INTERRUPT.h](#)
  - INTERRUPT\_Disable() : [INTERRUPT.h](#)
  - INTERRUPT\_Enable() : [INTERRUPT.h](#)
  - INTERRUPT\_GetActive() : [INTERRUPT.h](#)
  - INTERRUPT\_GetAppVersion() : [INTERRUPT.c](#) , [INTERRUPT.h](#)
  - INTERRUPT\_GetPending() : [INTERRUPT.h](#)
  - INTERRUPT\_Init() : [INTERRUPT.c](#) , [INTERRUPT.h](#)
  - INTERRUPT\_SetPending() : [INTERRUPT.h](#)
  - INTERRUPT\_STATUS : [INTERRUPT.h](#)
  - INTERRUPT\_STATUS\_FAILURE : [INTERRUPT.h](#)
  - INTERRUPT\_STATUS\_SUCCESS : [INTERRUPT.h](#)
  - INTERRUPT\_STATUS\_t : [INTERRUPT.h](#)
  - INTERRUPT\_t : [INTERRUPT.h](#)
-

# INTERRUPT

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- INTERRUPT\_ClearPending() : **INTERRUPT.h**
  - INTERRUPT\_Disable() : **INTERRUPT.h**
  - INTERRUPT\_Enable() : **INTERRUPT.h**
  - INTERRUPT\_GetActive() : **INTERRUPT.h**
  - INTERRUPT\_GetAppVersion() : **INTERRUPT.c , INTERRUPT.h**
  - INTERRUPT\_GetPending() : **INTERRUPT.h**
  - INTERRUPT\_Init() : **INTERRUPT.c , INTERRUPT.h**
  - INTERRUPT\_SetPending() : **INTERRUPT.h**
-

# INTERRUPT

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- INTERRUPT\_STATUS\_t : [INTERRUPT.h](#)
  - INTERRUPT\_t : [INTERRUPT.h](#)
-

# INTERRUPT

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- INTERRUPT\_STATUS : [INTERRUPT.h](#)
- 



# INTERRUPT

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- INTERRUPT\_STATUS\_FAILURE : [INTERRUPT.h](#)
  - INTERRUPT\_STATUS\_SUCCESS : [INTERRUPT.h](#)
-

# INTERRUPT

Home	
File List	Globals

# INTERRUPT.h

[Go to the documentation of this file.](#)

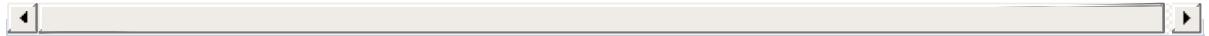
```
00001
00059 #ifndef INTERRUPT_H
00060 #define INTERRUPT_H
00061
00062 /*****
00063 ****
00064 * HEADER FILES
00065 ****
00066 ****
00067 ****
00068 #if (UC_SERIES == XMC14)
00069 #include <xmc_scu.h>
00070 #endif
00071
00072 #include "interrupt_conf.h"
00073
00074
00080 ****
00081 ****
00082 * MACROS
00083 ****
```

```
***** */
00083
00084 /*****
***** ****
***** ****
00085 * ENUMS
00086 ****
***** ****
***** ****
***** */
00094 typedef enum INTERRUPT_STATUS
00095 {
00096     INTERRUPT_STATUS_SUCCESS = 0U,
00097     INTERRUPT_STATUS_FAILURE = 1U
00098 } INTERRUPT_STATUS_t;
00103 ****
***** ****
***** ****
***** */
00104 * DATA STRUCTURES
00105 ****
***** ****
***** */
00114 typedef struct INTERRUPT
00115 {
00116 #if(UC_SERIES == XMC14)
00117     const XMC_SCU_IRQCTRL_t irqctrl;
00118 #endif
00119     const IRQn_Type node;
00120     const uint8_t priority;
00121 #if(UC_FAMILY == XMC4)
00122     const uint8_t subpriority;
00123 #endif
00124     const bool enable_at_init;
00125 } INTERRUPT_t;
00126
00131 ****
***** ****
***** ****
```

```
00132 * API PROTOTYPES
00133 ****
00134 ****
00135 #ifdef __cplusplus
00136 extern "C" {
00137 #endif
00138
00169 DAVE_APP_VERSION_t INTERRUPT_GetAppVersion(v
00169 oid);
00189 INTERRUPT_STATUS_t INTERRUPT_Init(const INTE
00189 RRUPT_t *const handler);
00190
00210 __STATIC_INLINE void INTERRUPT_Enable(const
00210 INTERRUPT_t *const handler)
00211 {
00212     XMC_ASSERT("Handler NULL", (handler != NUL
00212 L));
00213     NVIC_EnableIRQ(handler->node);
00214 }
00215
00235 __STATIC_INLINE void INTERRUPT_Disable(const
00235 INTERRUPT_t *const handler)
00236 {
00237     XMC_ASSERT("Handler NULL", (handler != NUL
00237 L));
00238     NVIC_DisableIRQ(handler->node);
00239 }
00240
00261 __STATIC_INLINE uint32_t INTERRUPT_GetPending
00261 (const INTERRUPT_t *const handler)
00262 {
00263     XMC_ASSERT("Handler NULL", (handler != NUL
00263 L));
00264     return NVIC_GetPendingIRQ(handler->node);
00265 }
```

```
00266
00286 __STATIC_INLINE void INTERRUPT_SetPending(const INTERRUPT_t *const handler)
00287 {
00288     XMC_ASSERT("Handler NULL", (handler != NULL));
00289     NVIC_SetPendingIRQ(handler->node);
00290 }
00291
00324 __STATIC_INLINE void INTERRUPT_ClearPending(const INTERRUPT_t *const handler)
00325 {
00326     XMC_ASSERT("Handler NULL", (handler != NULL));
00327     NVIC_ClearPendingIRQ(handler->node);
00328 }
00329
00330 #if(UC_FAMILY == XMC4)
00331
00352 __STATIC_INLINE uint32_t INTERRUPT_GetActive(const INTERRUPT_t *const handler)
00353 {
00354     XMC_ASSERT("Handler NULL", (handler != NULL));
00355     return NVIC_GetActive(handler->node);
00356 }
00357
00358 #endif
00359
00360 #ifdef __cplusplus
00361 }
00362 #endif
00363
00364 #include "interrupt_extern.h"
00365
00366 #endif
00367
```





# INTERRUPT

Home

File List

Globals

## INTERRUPT.c

Go to the documentation of this file.

```
00001
00055 /*****
***** 
***** 
00056 * HEADER FILES
00057 ****
***** 
***** 
***** /
00058
00059 #include "interrupt.h"
00060
00061 /*****
***** 
***** 
00062         * MACROS
00063 ****
***** 
***** 
***** /
00064
00065 /*****
***** 
***** 
***** 
00066 * LOCAL DATA
00067 ****
***** 
***** 
***** /
00068
```

```
00069 /*****  
*****  
*****  
00070 * LOCAL ROUTINES  
00071 *****  
*****  
*****  
*****  
*****  
00072  
00073 /*****  
*****  
*****  
*****  
00074 * API IMPLEMENTATION  
00075 *****  
*****  
*****  
*****  
*****  
00076 /*  
00077 * API to retrieve the version of the INTERRUPT APP  
00078 */  
00079 DAVE_APP_VERSION_t INTERRUPT_GetAppVersion(v  
oid)  
00080 {  
00081     DAVE_APP_VERSION_t version;  
00082  
00083     version.major = INTERRUPT_MAJOR_VERSION;  
00084     version.minor = INTERRUPT_MINOR_VERSION;  
00085     version.patch = INTERRUPT_PATCH_VERSION;  
00086  
00087     return (version);  
00088 }  
00089  
00090 /*  
00091 * API to initialize the INTERRUPT APP  
00092 */  
00093 INTERRUPT_STATUS_t INTERRUPT_Init(const INTE  
RRUPT_t *const handler)  
00094 {
```

```
00095     XMC_ASSERT("INTERRUPT_Init:HandlePtr NULL"
, (handler != NULL));
00096
00097 #if(UC_FAMILY == XMC4)
00098
00099     NVIC_SetPriority(handler->node,
00100                     NVIC_EncodePriority(NVIC_
GetPriorityGrouping(),
00101                                         handl
er->priority,
00102                                         handl
er->subpriority));
00103     if (handler->enable_at_init == true)
00104     {
00105         INTERRUPT_Enable(handler);
00106     }
00107 #endif
00108
00109 #if(UC_FAMILY == XMC1)
00110     NVIC_SetPriority(handler->node, handler->p
riority);
00111
00112 #if (UC_SERIES == XMC14)
00113     XMC_SCU_SetInterruptControl((uint8_t)handl
er->node, (XMC_SCU_IRQCTRL_t)((handler->node << 8)
| handler->irqctrl));
00114 #endif
00115
00116     /* Enable the interrupt if enable_at_init
is enabled */
00117     if (handler->enable_at_init == true)
00118     {
00119         INTERRUPT_Enable(handler);
00120     }
00121 #endif
00122
00123     return (INTERRUPT_STATUS_SUCCESS);
```

00124 }

---